Bachelor of Software Engineering - Game Programming

GD2P02 – Physics Programming

Collision Detection - Part 2

Overview

- Collision Detection
  - Discrete Collision Detection
    - Methods and Approaches
      - Narrow Phase
      - Separating Axis Theorem
  - Point in a Triangle
  - Barycentric Coordinates
  - Exercises

Narrow Phase continued…

- Swap and change collision detection algorithms…
  - Remember there are often multiple ways of solving a problem!
- Create a library of solutions to solve different problems…
  - Reuse!
- Develop and test!
  - Compare and contrast algorithms…
  - Do unit testing!

Narrow Phase continued…

- What collision data is needed?
  - Have two bodies collided?
  - At what point are the two bodies colliding?
  - What is the normal of the collision?
    - Direction that would push the two objects out of penetration the quickest…
  - What is the penetration depth?
- Rigid bodies in the real world do not interpenetrate.
  - These real-world interactions have to be emulated.
- Ensure objects don't stick, jiggle or explode.

Intersection approach

- Intersection testing is based on the theorem:

"Two convex polytopes are disjoint iff there exists a separating axis orthogonal to a face of either polytope or orthogonal to an edge from each polytope."

[Gottschalk et al. '96]

  - Find a plane that separates the two objects.
    - If a plane can be found, the objects are not colliding.

Separating Axis Theorem

Two phases in SAT:

1) Has a collision occurred and what is the best plane for separation?

2) Use plane to extract contact information.

First:

- Project all points of a shape onto a plane.
- Project both objects onto the same plane…
  - Now it is a 2D overlap test.

Separating axis: An axis on which the projections of two polytopes do not overlap.

Separating Axis Theorem continued…

- Testing two shapes that are not intersecting
  - A line can be drawn between these two shapes, which shows they are not intersecting.

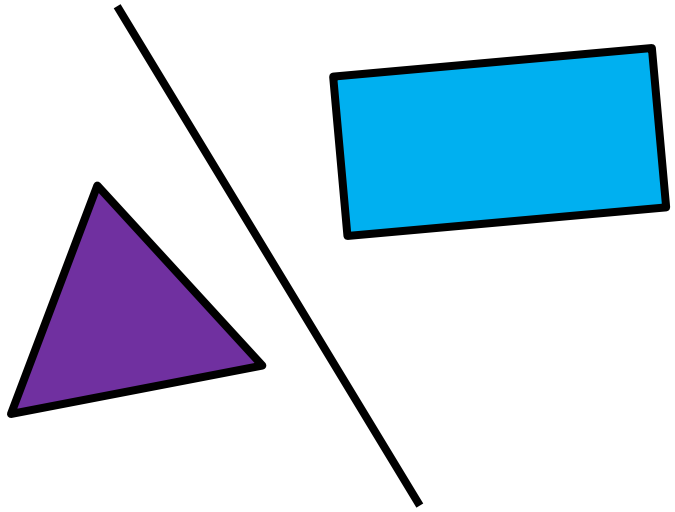

Fig 9: Line separating the triangle and the rectangle.

Separating Axis Theorem continued…

- Taking a perpendicular line to the separation line…
  - Projecting the shapes onto the new line…
  - There is no overlap in the projections!
  - The first axis found where the projections do not overlap determines there is no intersection!

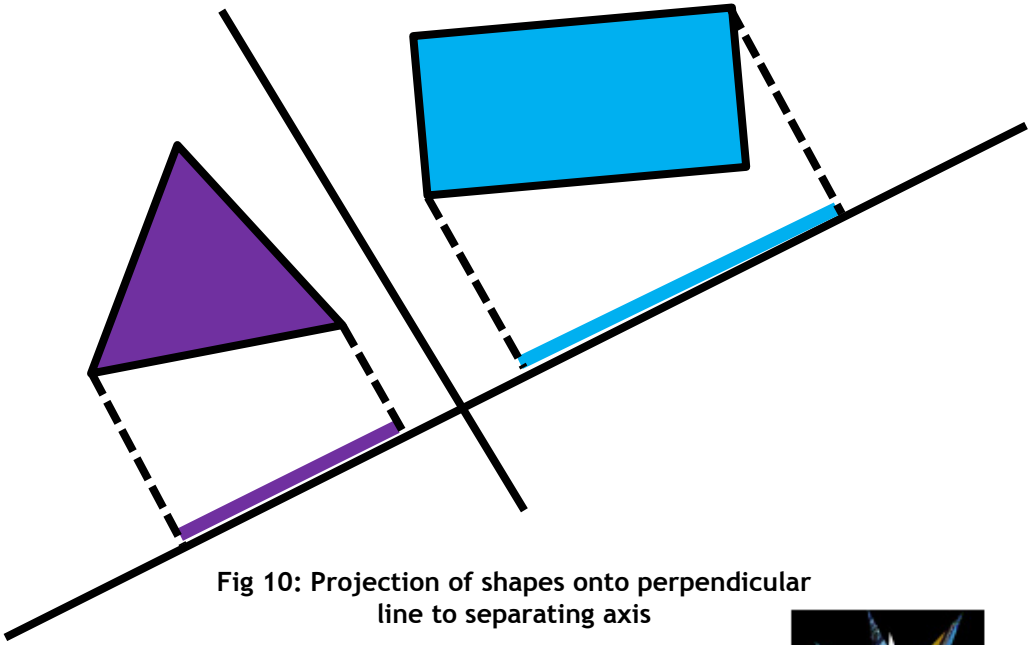

Fig 10: Projection of shapes onto perpendicular line to separating axis

Separating Axis Theorem continued…

- If for all axes, the shape projections overlap…
  - Then we know the shapes are intersecting.
  - All axes must be tested for overlap to determine intersection.
  - But what axes do we test?

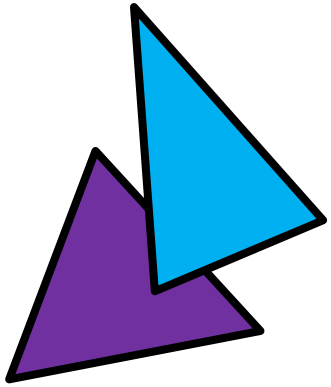

Fig 11: Overlapping triangles.

Separating Axis Theorem: Finding the Axes to Test

- The axes to test are the normal of each shapes edges!
- Trick:
  - Flip the coordinates of each edge, negate one of them…
  - For example:
    - (x, y)
  - Result:
    - (-y, x)
  - Or:
    - (y, -x)

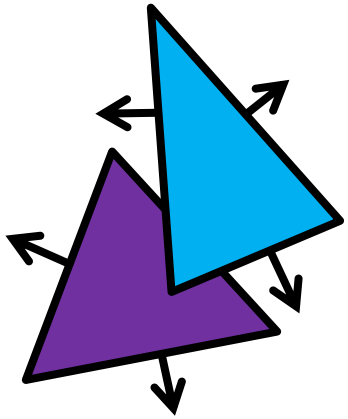

Fig 12: Normals to edges of each shape.

Separating Axis Theorem: Considerations…

- Reduce the number of axes to test against:
  - Do not test parallel axes…
    - A rectangle only has two axes to test.
- SAT in 3D can end up with many angles to test!
  - Think about every edge added to a 3D piece of geometry…
- The more collisions there are, the worse the algorithm performs:
  - Early out as soon as one separating axis is detected…
  - Otherwise must check all!

Separating Axis Theorem: Considerations…

- It can be used on concave polygons.
  - But the polygon must first be broken up into convex polygons…
  - So extra logic must be conducted!
    - Object Hierarchy

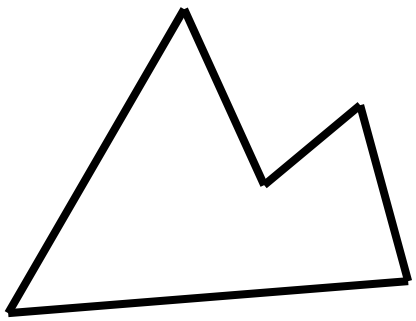

Fig 13: Single concave polygon.

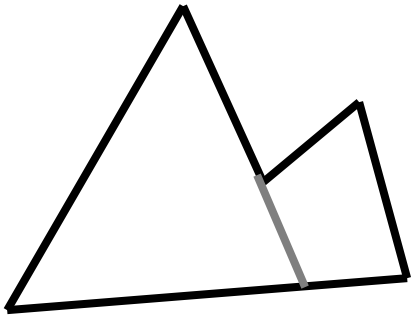

Fig 14: Single concave polygon broken into two convex polygons.

Concave Polygon

- If any internal angles are greater than 180 degrees, then the polygon is concave.

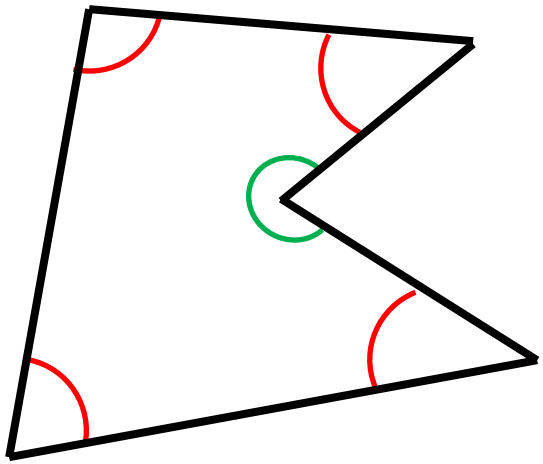

Fig 8: Concave polygon, green angle is greater than 180 degrees.

Point in a Triangle

- How can we check if a point is in a triangle?
- Dot product angle method.
  - Calculate the dot product between a point on a triangle's plane and the triangle's three corners, and get the angle
  - The total sum of the angles inside the triangle from the point will add up to 360 degrees.
  - If the value is not equal to 360 the point is outside the triangle.

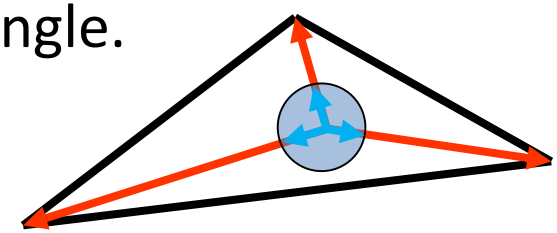

Fig 15: Point inside a triangle,
red vector = (triangle point - point),
blue vector = normalised(red vector)

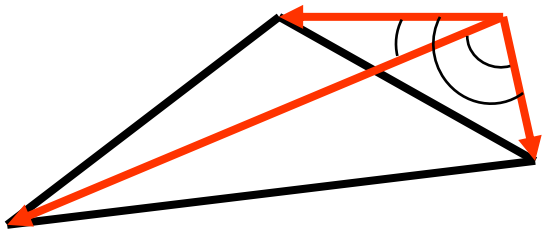

Fig 16: Point outside a triangle.

Barycentric Coordinates

- Weighted coordinate system.
- Vertices within a shape are given a weighting.
  - Based upon position.
  - Single Law:
    - The sum of the weights must add up to one!
- Applied to: lines, triangles…
  - Find if two triangles are intersecting!

Barycentric Coordinates continued…

- Parametric representation for a line:

$$P = A + t(B - A)$$
$$= (1 - t)A + tB$$
$$= uA + vB$$

  - A and B are the start and end points of the line.
  - P is a point on the line where t is from 0 to 1.
- u and v are Barycentric coordinates…
  - For a point on the line segment.
  - Conditions:
    - $u + v = 1$
    - $0 \leq u \leq 1$ and $0 \leq v \leq 1$

Barycentric Coordinates continued…

- Representing a triangle using Barycentric coordinates:
  - A point that can be anywhere within the edges of a triangle based upon three weights.
  - Three vertices of a triangle A, B, C and the point P:

    $P = uA + vB + wC$

    - $u + v + w = 1$
    - $0 \leq v \leq 1$
    - $0 \leq w \leq 1$
    - $v + w \leq 1$

Barycentric Coordinates continued…

- Simplify the equation so that it only depends on two weights:

$$P = uA + vB + wC = (1 - v - w)A + vB + wC$$
$$= A + v(B - A) + w(C - A)$$

  - The point P is represented by two scalar weights, v and w.
- Since there are three vertices on the same plane:
  - $P = A + v(B - A) + w(C - A)$
  - $v(B - A) + w(C - A) = P - A$
  - $vv_0 + wv_1 = r$
  - $v\,(v_0 \times v_0) + w\,(v_0 \times v_1) = (v_0 \times r)$
  - $v\,(0) + w\,(v_0 \times v_1) = (v_0 \times r) \rightarrow w = (v_0 \times r) / (v_0 \times v_1)$

Barycentric Coordinates continued…

- Steps for solution:
  - Pick a vertex of the triangle to be the local origin
  - Compute the vectors of the triangle by using the vertices
    - v_0 = B-A, v_1 = C-A
  - Compute the barycentric values for the given point P with respect to the local origin
  - Check whether v and w are both within the constraints.
    - If so, return true. Otherwise, if the point is outside, return false.

Barycentric Coordinates continued…

- Barycentric Coordinates on an equilateral triangle:

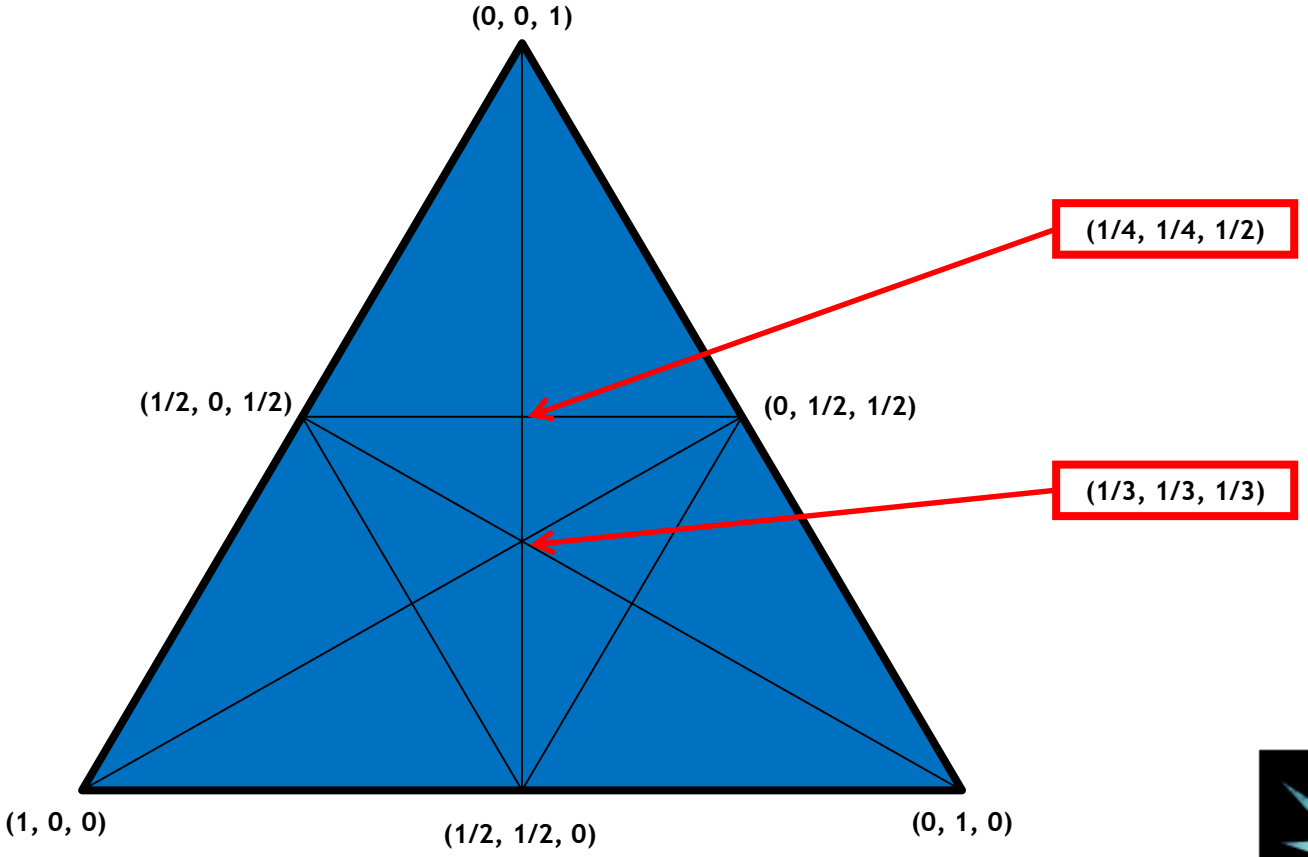

Fig 17: Barycentric coordinates on an equilateral triangle.

Exercise 004.1 - Point in Triangle Application

- Define a triangle:
  - User selects three points.
  - Point and click.
- Define a point:
  - User selects one point.
  - Point and click.
- Render the triangle and point to test.
- Determine if the point is in the triangle or not.
- Render the result: In or Out?
- Allow the scene to be reset:
  - The R key.

Exercise 004.2 - Barycentric Coordinate Calculator

- Define a triangle:
  - User selects three points.
- Define a point:
  - User selects one point.
- Calculate the Barycentric coordinate of the point to find if the point is on the triangle.
- Allow the user to select a new point.
  - Point and click.
- Reset the scene:
  - The R key.

Exercise 004.3 - Separating Axis Theorem

- Allow the user to define two convex polygon shapes.
  - Clicking to define points
  - Render the shapes.
- Use SAT to determine if the two shapes intersect.
  - Report the results to the user.
- Allow the scene to be reset:
  - The R key.

Summary

- Collision Detection
  - Discrete Collision Detection
    - Methods and Approaches
      - Narrow Phase
      - Separating Axis Theorem
  - Point in a Triangle
  - Barycentric Coordinates
  - Exercises